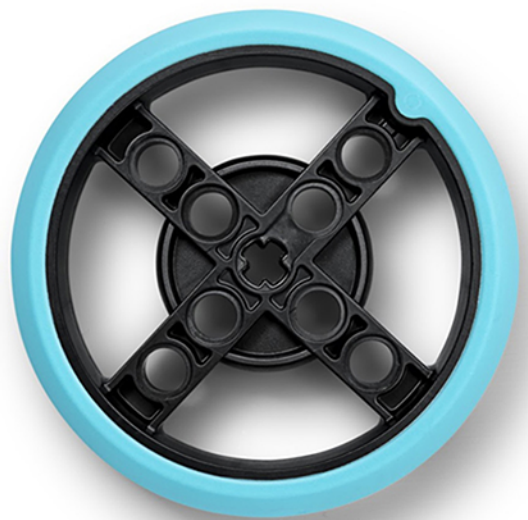


*Let's roll*  
**SPIKE**



**Dimitrios Kravvaris**

Unofficial Lego Education SPIKE Prime eBook



ISBN: 978-618-00-1730-4



# Let's roll SPIKE

Author: **Dimitrios Krawvaris**, Ph.D. Information Engineer

Computer Science teacher & STEM trainer.

Email: [jkravv@gmail.com](mailto:jkravv@gmail.com)

ISBN: 978-618-00-1730-4



Ioannina, Greece

2020



The role of the teacher is to create the conditions for invention rather than provide ready-made knowledge.

*Seymour Papert*



## Table of content

1.	Introduction.....	3
2.	Motors .....	5
2.1	Types.....	5
2.1.1	How to use motors.....	5
2.1.2	Simple vehicle (building Instructions).....	6
2.2	Distance .....	12
2.2.1	Wheels.....	12
2.2.2	Movement.....	12
2.2.3	Motor degrees and vehicle distance .....	13
2.3	Turns .....	14
2.3.1	Go around.....	14
2.3.2	Square track .....	15
3.	Sensors .....	17
3.1	Distance Sensor .....	17
3.1.1	Code blocks for distance sensor .....	17
3.1.2	Distance sensor add-on (building instructions).....	17
3.1.3	Slow down .....	20
3.2	Force sensor .....	21
3.2.1	Code blocks for force sensor.....	21
3.2.2	Force sensor add-on (building instructions).....	21
3.2.3	How hard? .....	24
3.3	Color sensor.....	25
3.3.1	Code blocks for color sensor .....	25
3.3.2	Color sensor add-on (building instructions) .....	25
3.3.3	Count lines.....	28
3.3.4	Follow line (P Controller) .....	29
3.4	Gyro sensor.....	31
3.4.1	Code blocks for gyro sensor .....	31
3.4.2	Go straight ... no matter what! .....	31





## 1. Introduction

This book is an unofficial Lego Education SPIKE guide for Lego WeDo users, Scratch programmers and new users of educational robotics. Its purpose is to familiarize the readers with the active elements of SPIKE and through simple activities to stimulate exploration and creativity. It focuses on the movement of robotic vehicles by adding to each section new programming and operating elements. The readers examine that way the programming environment and the operation of the robot one step at the time.

More specifically, each section contains information about SPIKE elements as provided by the Lego site [*Lego information*] as well as related photos. In addition, instructions are given to build the robotic vehicle and the sensors add-ons. Moreover, there is a task (or two) in every section that requires a program in order the robot to perform a specific process. We provide the code and comments on how it works. We encourage the reader to experiment further by acquiring knowledge through practice. It should be emphasized that programming in the Scratch environment can provide more than one solution to every problem, so the code we provide in each case is indicative.

Through constructive activities, trainees will gain confidence that they can move intelligent their robotic vehicle by adding a “brick” of knowledge in every section.



## 2. Motors

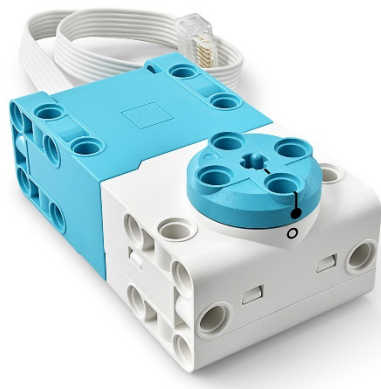
### 2.1 Types

#### A) *Medium Angular Motor*



Build high-response robots with the LEGO® Technic™ Medium Angular Motor featuring low-profile design, integrated rotation sensor with absolute positioning and 1-degree accuracy [Lego information].

#### B) *Large Angular Motor*



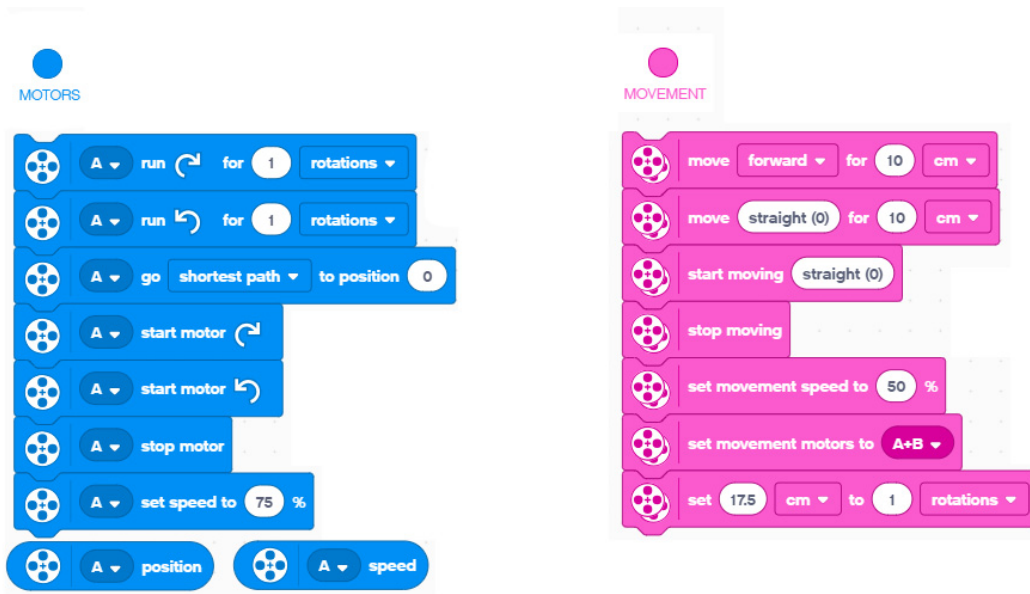
The LEGO® Technic™ Large Angular Motor is the ideal solution for high-power, high-torque applications, featuring an integrated rotation sensor and absolute positioning for true straight-line control [Lego information].

Each motor is designed to function in as both a motor and sensor. With the integrated advanced Rotation Sensor, the motor can report both speed and position. The motor can also sense direct user input measures if the output is rotated by hand [Lego Information]. The basic SPIKE Prime kit contains two medium motors and a large one.

#### 2.1.1 How to use motors

The motors are connected to any port on the Hub (A - F). In our case they will be combined with the wheels in the kit and we will start rolling!!!

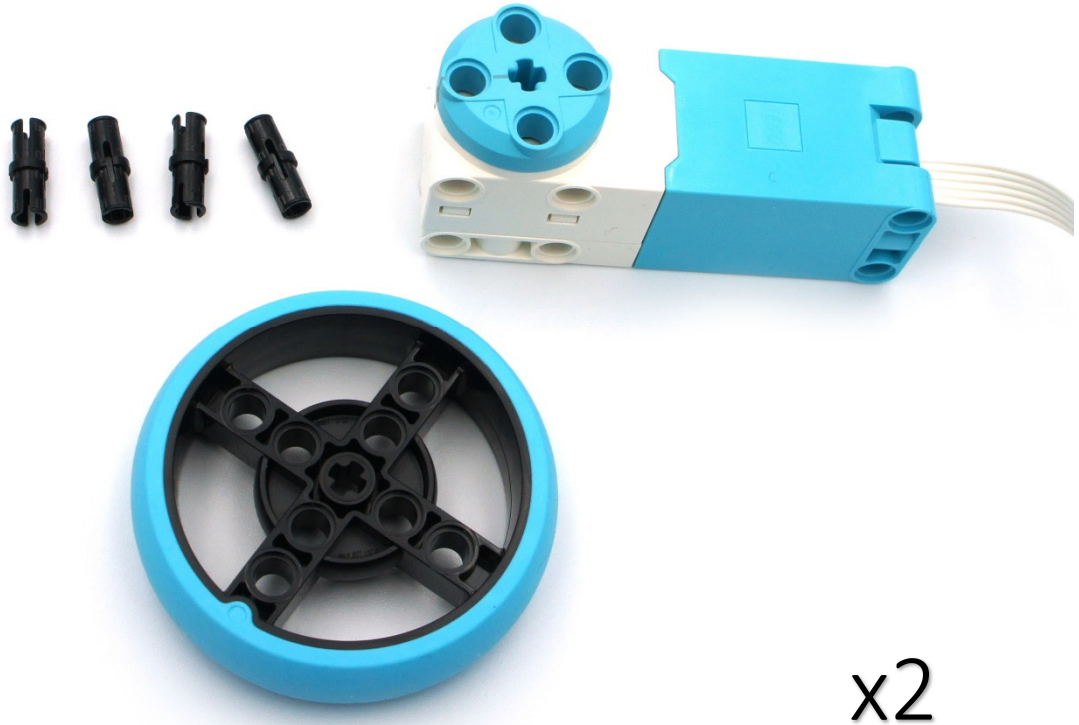
The basic commands for Motors and Movement of the programming environment are presented in the following figure, and the examples that follow in the next paragraphs will show how to use them.



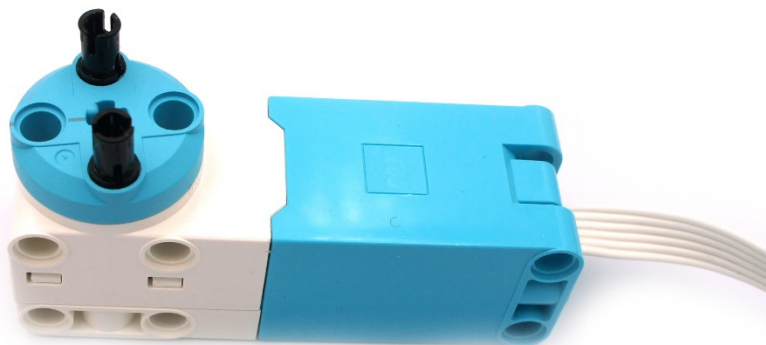
### 2.1.2 Simple vehicle (building Instructions)



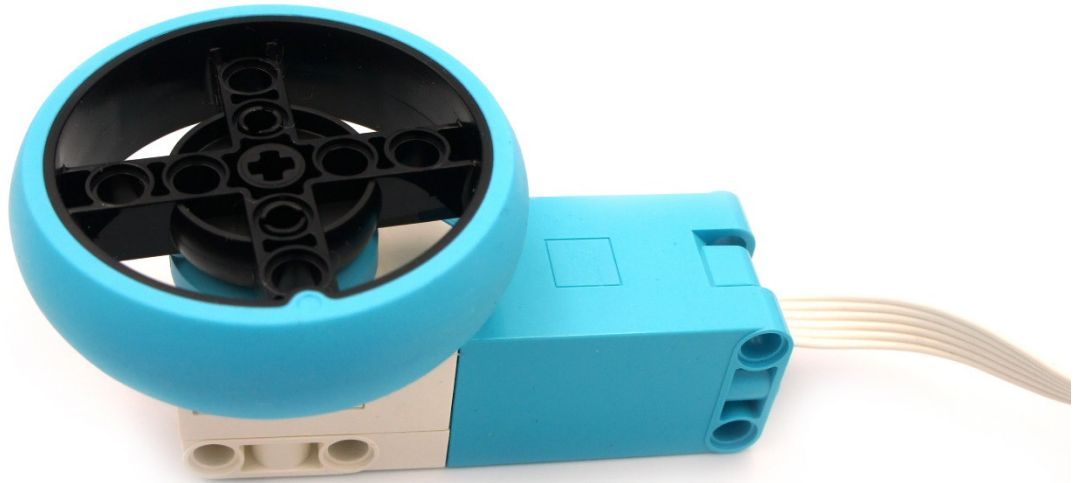




x2



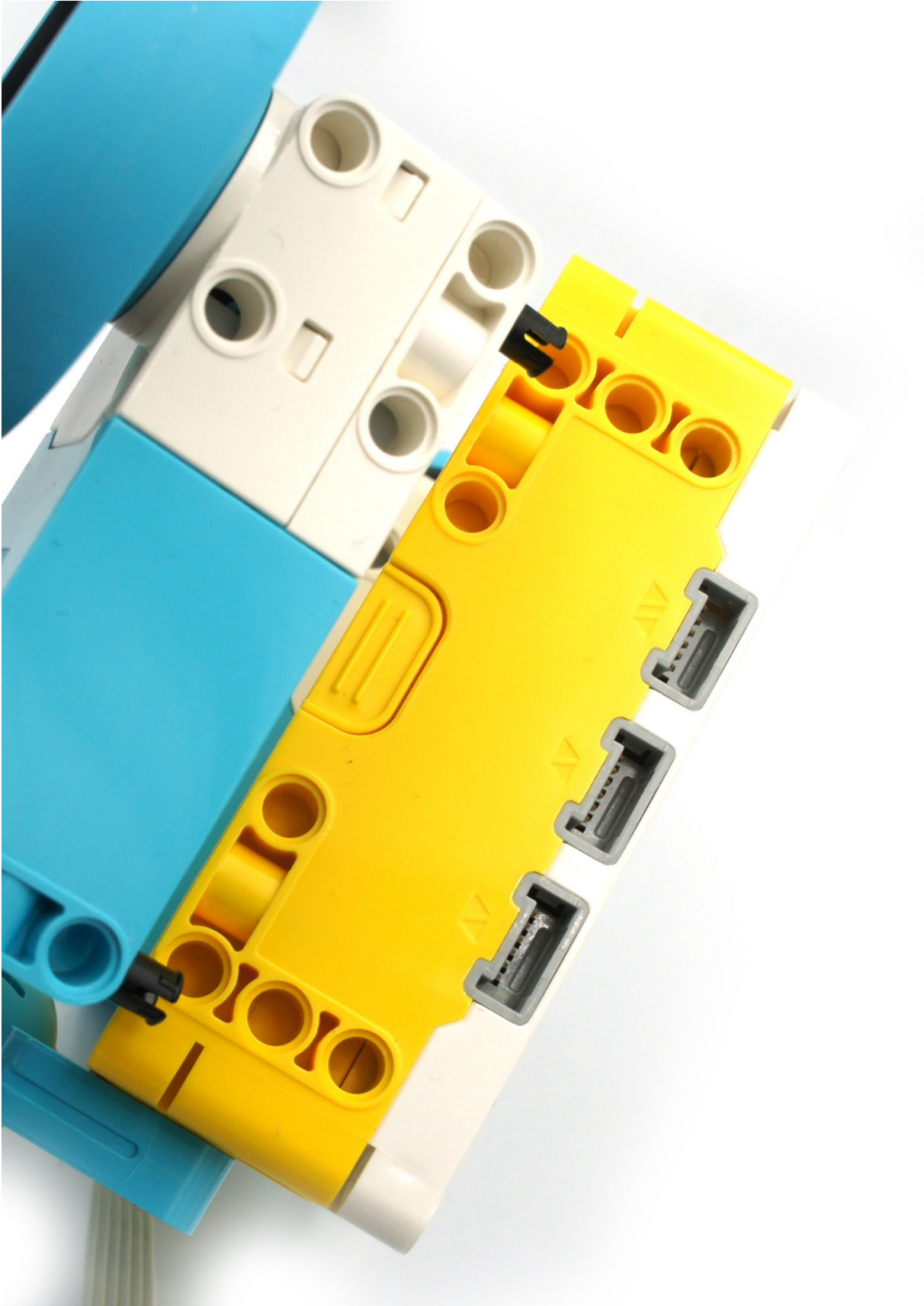
x2



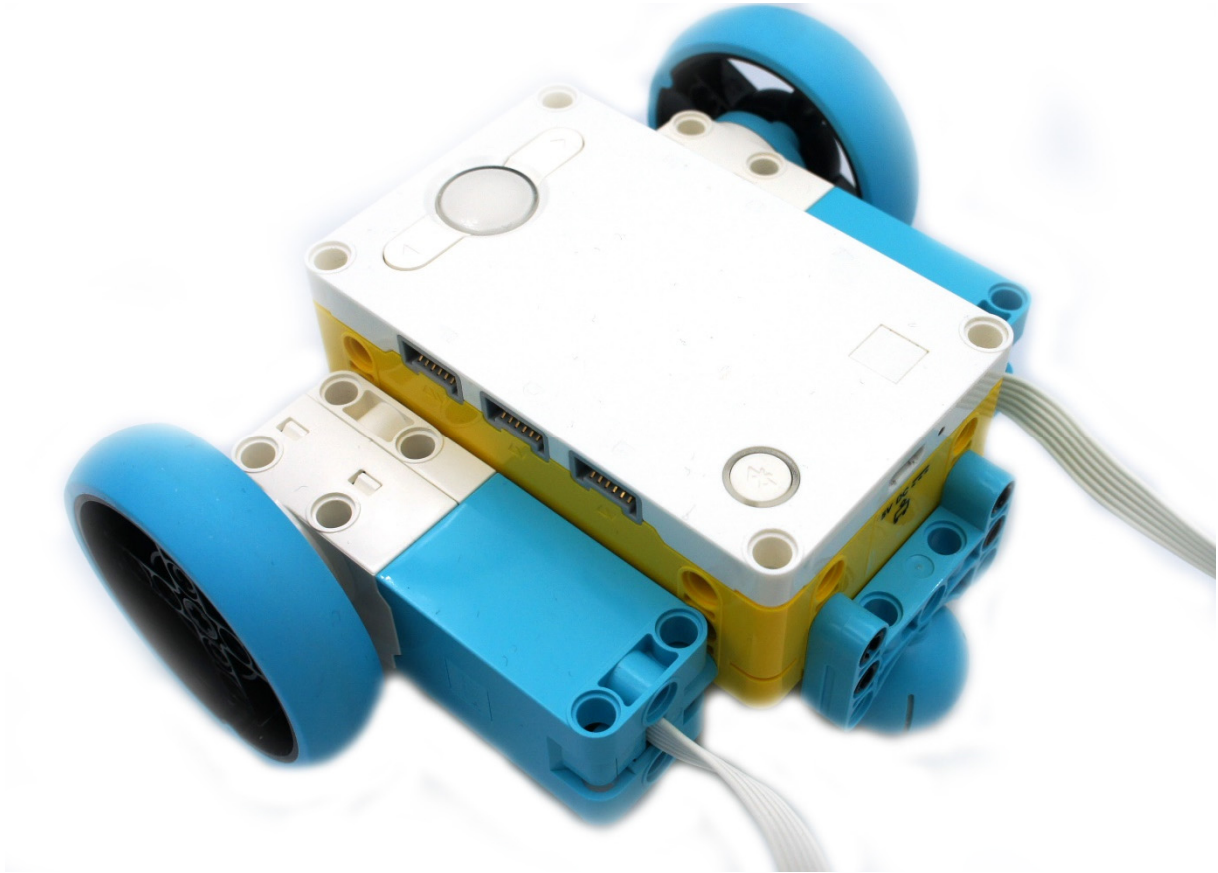
x2



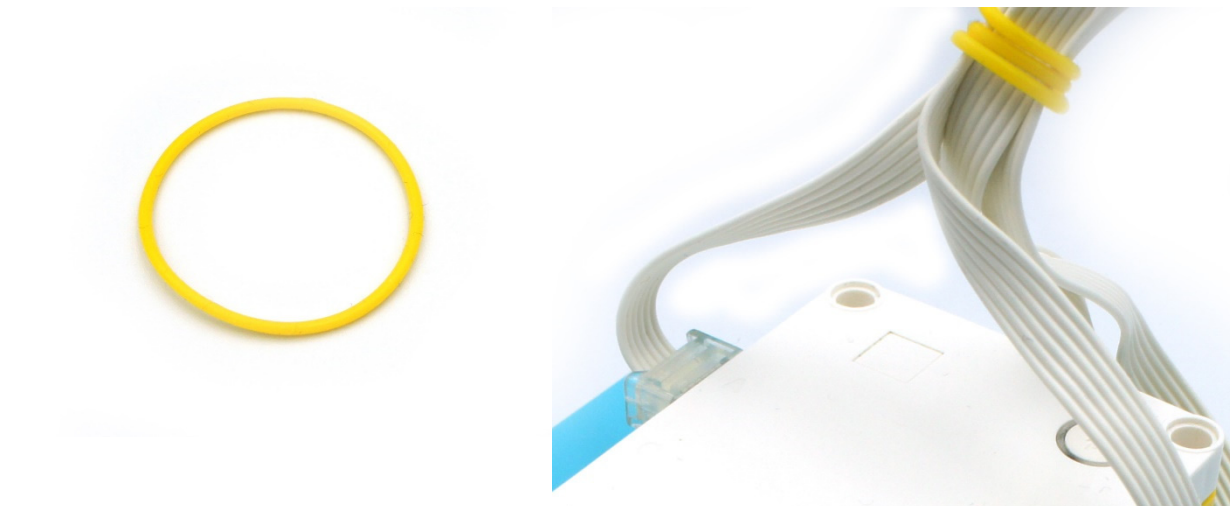
x2







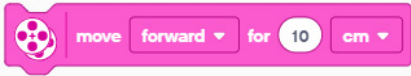
Wire Management



## 2.2 Distance

### 2.2.1 Wheels

The distance that our vehicle will move depends on two interdependent factors. The first is the motion of the motors (degrees, time etc.) and the second is the size of the wheels of the vehicle. For the same motors movement the vehicle with the largest wheels travels the longest distance. We know that a complete rotation of the motor will make the vehicle travel as far as its wheel circumference, i.e.  $2\pi r$  ( $\pi = 3.14$  and  $2r = \text{diameter of the wheel}$ ). So for the new SPIKE wheel the distance is  $3.14 * 56\text{mm} \sim 176\text{mm}$  for 1 motor rotation (or 360 degrees).

The command  is predefined by Lego for this wheel dimension, so the distance is calculated automatically.



### 2.2.2 Movement

Make the vehicle move forward for 15 cm.

Executing the following code the vehicle moves at a predetermined distance (we can confirm the result by counting with a ruler).



Set the motors ports.

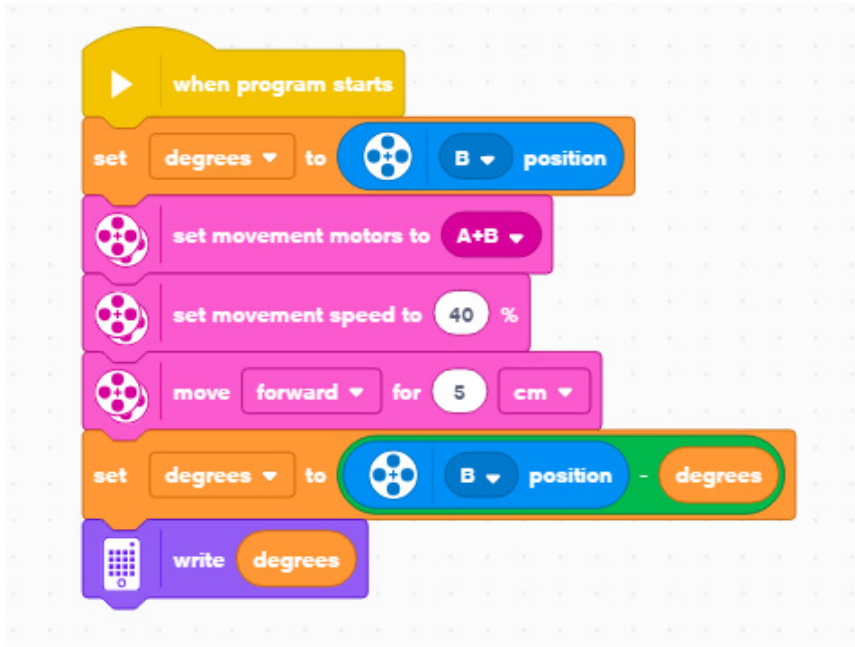
Set the motors power.

Set the direction (**forward/backward**) the number (**15**) and the type of measurement (**cm, inches, seconds, degrees, rotations**).

### 2.2.3 Motor degrees and vehicle distance

How many degrees do I have to adjust the motors to move the vehicle forward 5cm?

Executing the following code, the vehicle moves forward 5cm, the sensor measures the degrees it has traveled and finally displays it on the brick's screen.



Create a variable named degrees and set the value of the position of B Motor.

When vehicle complete the 5cm track set to degrees the new B Motor position minus the initial one. Show the result on the brick's screen.

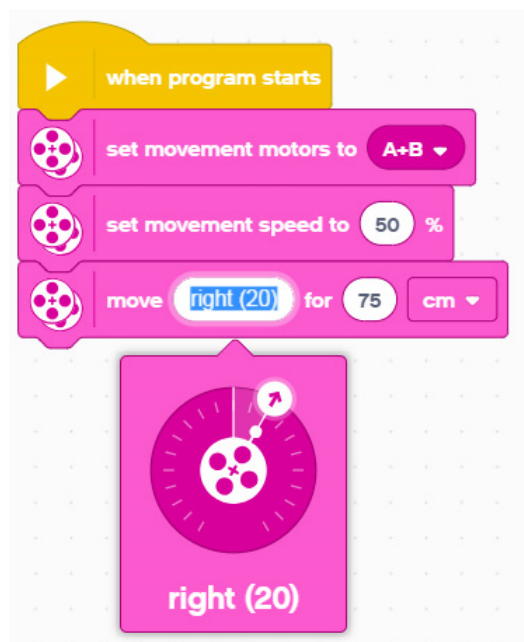
## 2.3 Turns

### 2.3.1 Go around

Make a program that will help the vehicle go around a box.

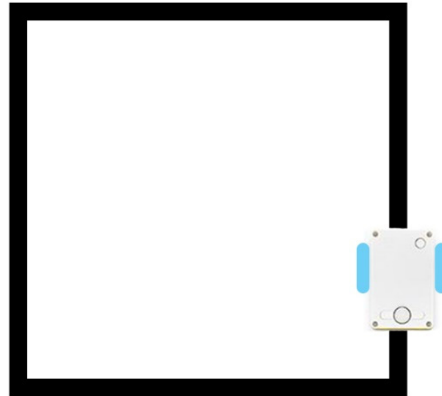


Execute the following code changing the **turn value (right or left)** and the **distance value (cm)** in order make a full track.

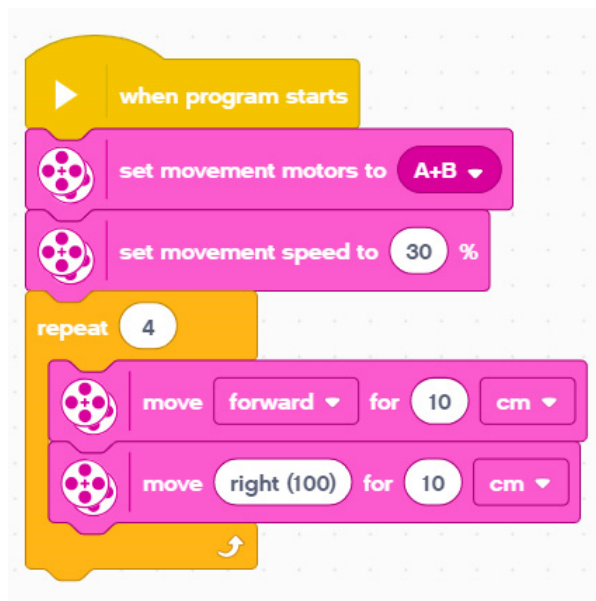


### 2.3.2 Square track

Make a program that will help the vehicle to follow a perfect 10cm X 10cm square track.



Execute the following code. The distance value of the **move right (100)** block give to our model a 90 degrees turn.





### 3. Sensors

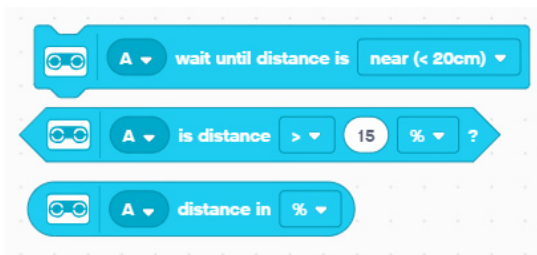
#### 3.1 Distance Sensor



Deliver high-accuracy results with the LEGO® Technic™ Distance Sensor, featuring -200cm range, +/- 1cm accuracy, programmable LED 'eyes' and an integrated 6-pin adaptor for third-party sensors, boards and DIY hardware.

The sensor can measure the distance to an object or surface using ultrasonic technology. In addition, the sensor has a light output around the "eyes," which is divided into four segments that can be activated individually. The back of the sensor can be removed and used as "break-out" access to the LPF2 wired platform for third-party suppliers and advanced users [Lego Information].

##### 3.1.1 Code blocks for distance sensor

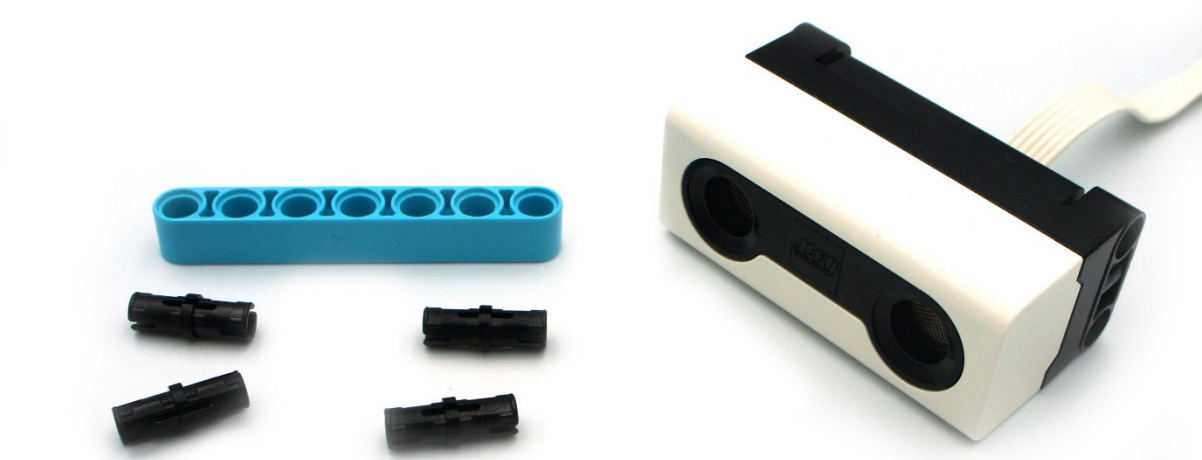


-*Command*. The block stops the execution of the program until distance is near, far or changed.

-*Condition*. Depending on conditions returns True or False.

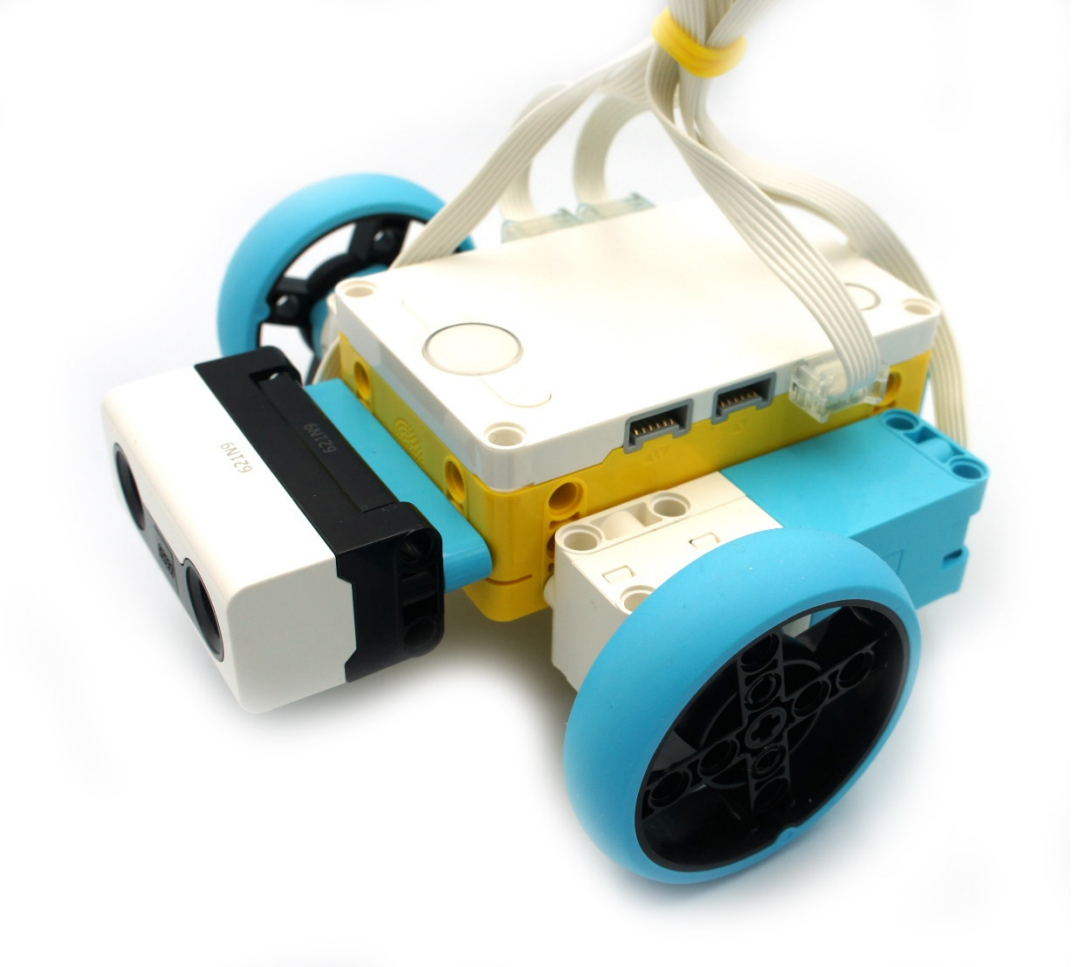
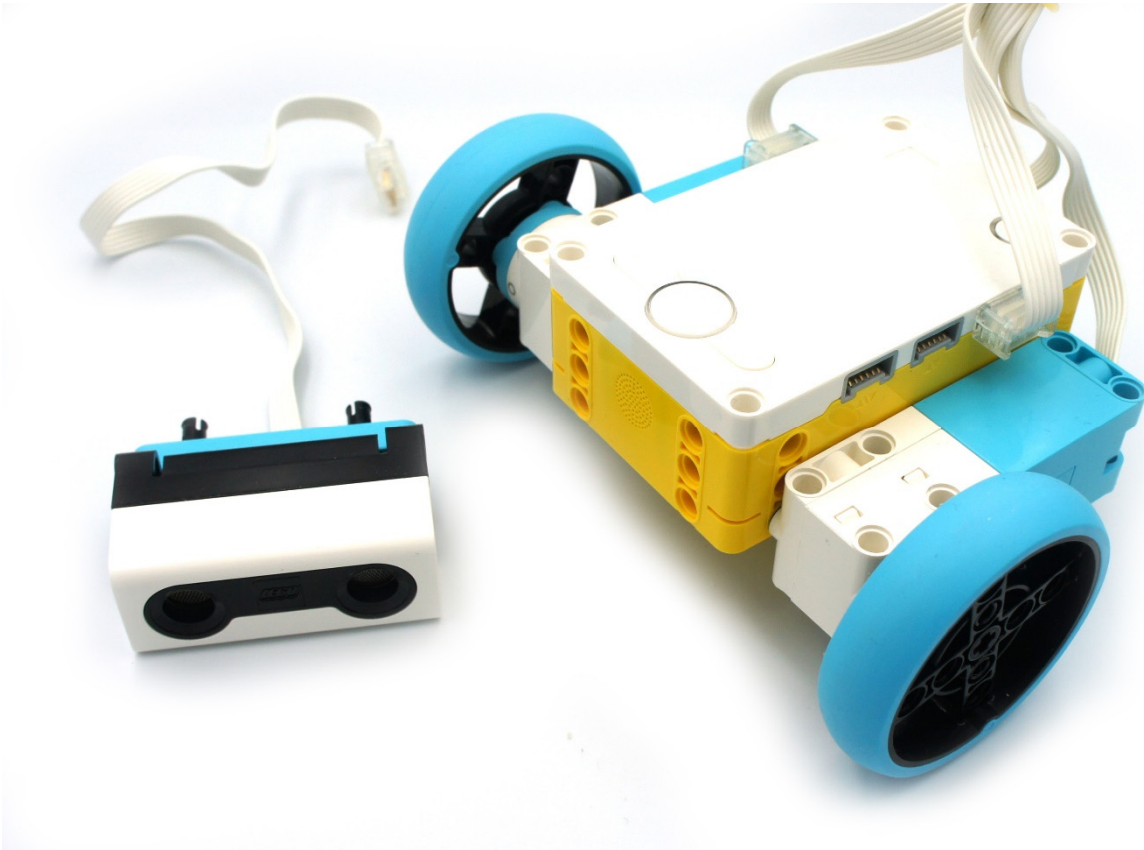
-*Value*. Returns the value of distance in %, cm or inches.

##### 3.1.2 Distance sensor add-on (building instructions)









### 3.1.3 Slow down

Make the vehicle to slow down as it approaches a

Execute the following code.

```
when program starts
  set movement motors to A+B
  forever loop
    set movement speed to C distance in % %
    start moving straight (0)
```

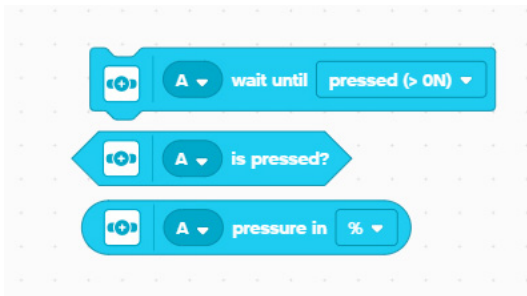
The sensor's value

### 3.2 Force sensor



The LEGO® Technic™ Force Sensor measures pressures of up to 10 Newtons (~1kg) for accurate, repeatable results. The sensor can also be used as a touch sensor when pressed, released or bumped [Lego Information].

#### 3.2.1 Code blocks for force sensor



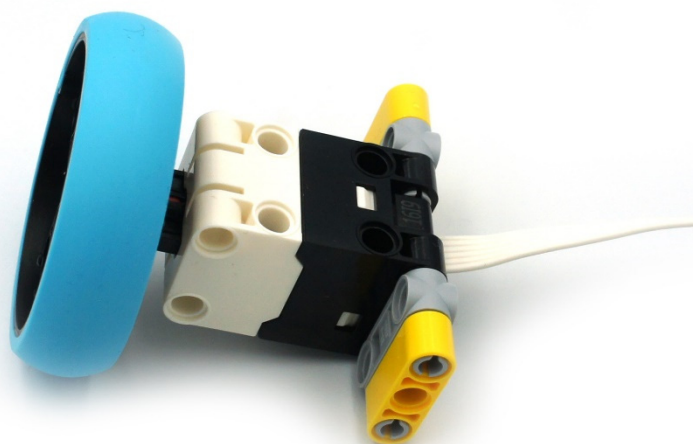
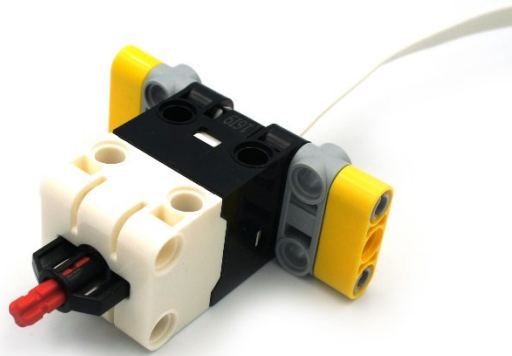
-*Command*. The block stops the execution of the program and continues depending the pressing mode value.

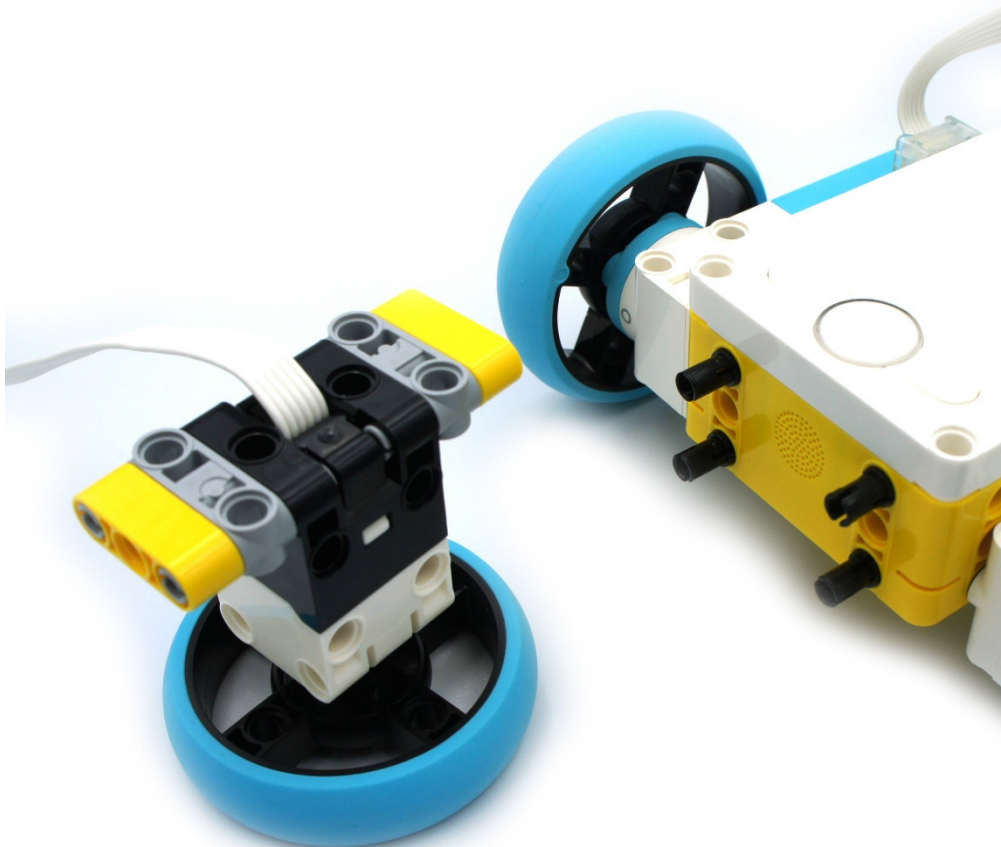
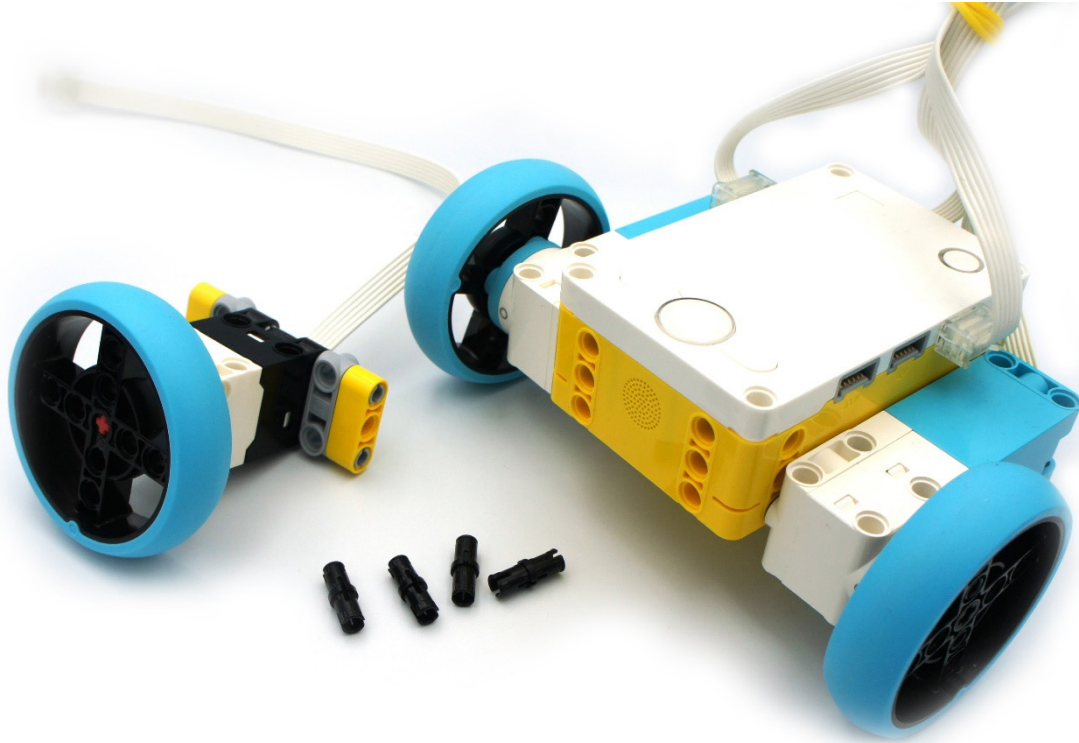
-*Condition*. If the force sensor is pressed it returns True.

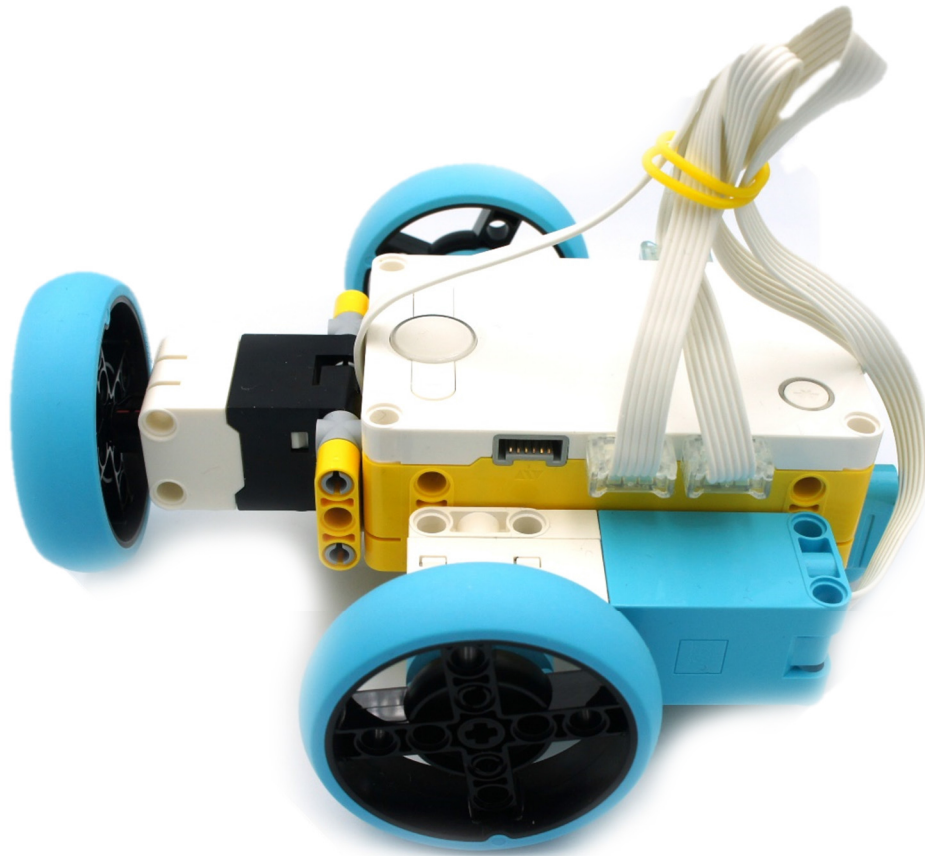
-*Value*. Returns the value of pressure in % or newton.

#### 3.2.2 Force sensor add-on (building instructions)





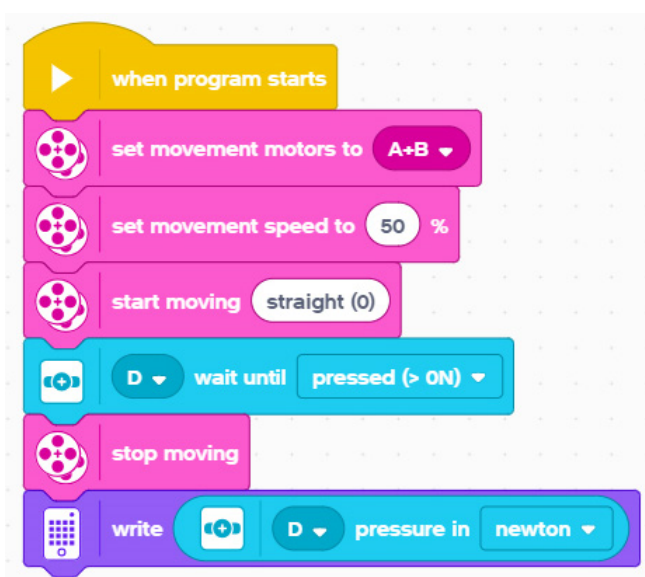




### 3.2.3 How hard?

Make a program that calculates the force of a vehicle crash on an obstacle and show the result on brick screen (use different values for the vehicle's speed).

Execute the following code. Test it for different speed values.



The vehicle starts moving

Until it hits the obstacle.

Then it stops.

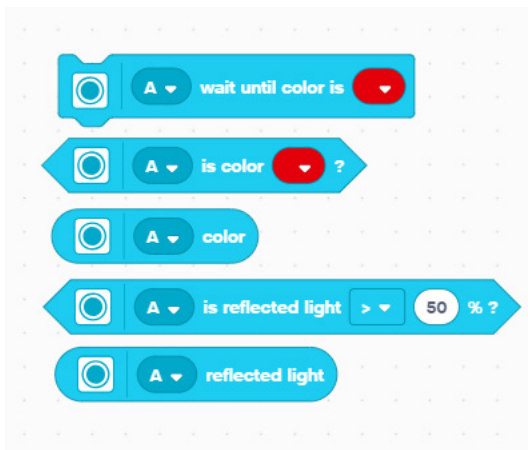
The force of the hit is displayed on bricks' screen.

### 3.3 Color sensor



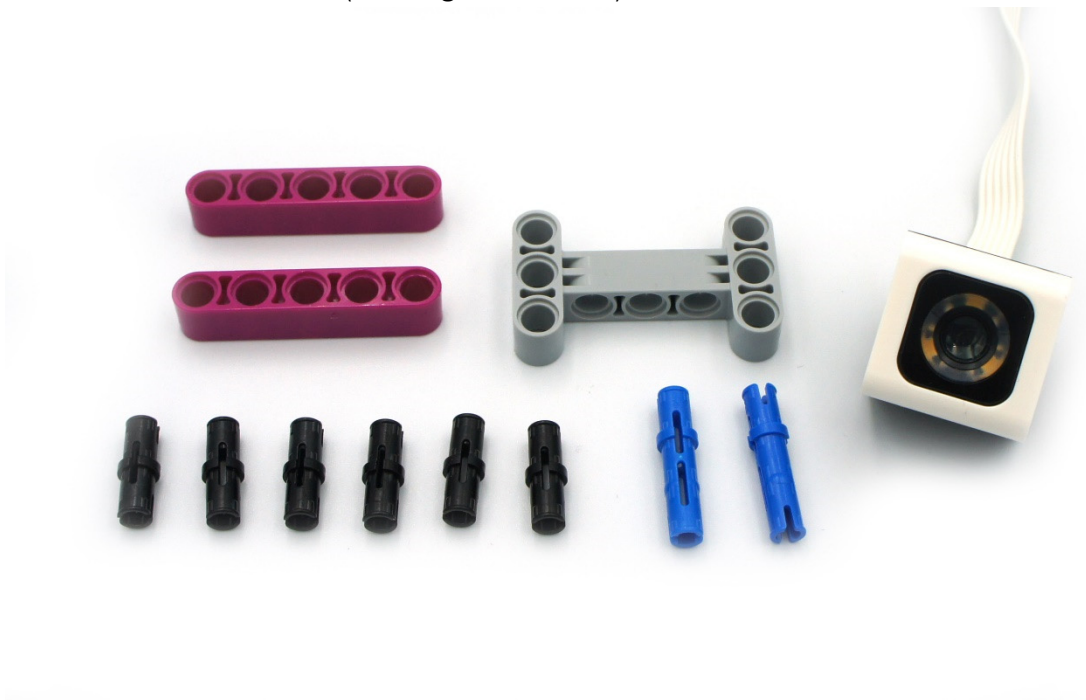
The LEGO® Technic™ Color Sensor distinguishes between 8 colors and measures reflected and ambient light from darkness to bright sunlight [Lego Information].

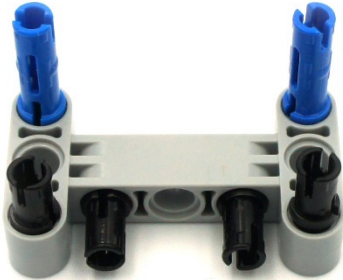
#### 3.3.1 Code blocks for color sensor



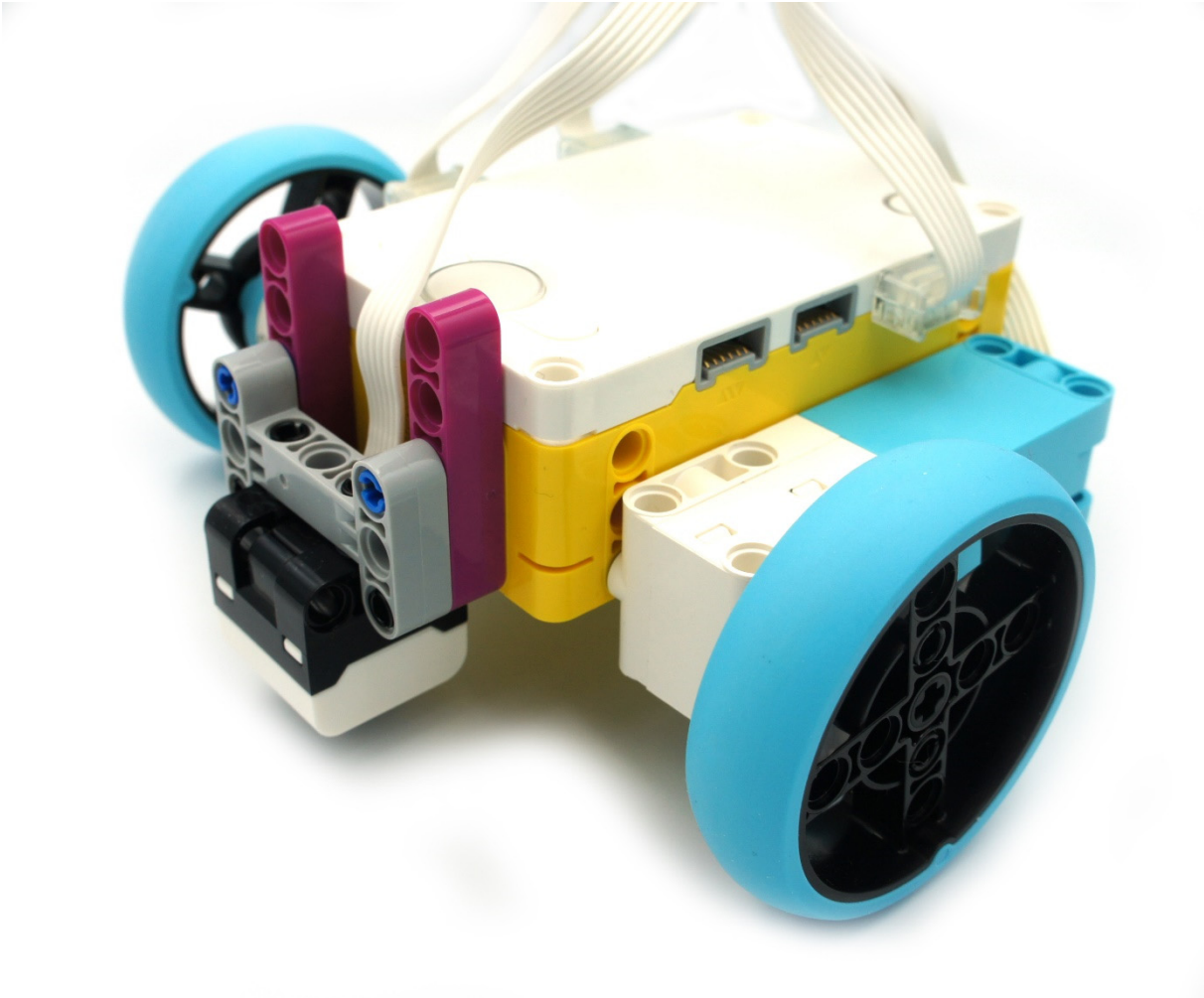
- Command*. The block stops the execution of the program and continues depending the value of the sensor (9 options available).
- Condition*. If the color sensor reads red it returns True.
- Value*. Returns the value of the color.
- Condition*. If the reflected light is higher than 50% it returns True.
- Value*. Returns the value of the reflected light.

#### 3.3.2 Color sensor add-on (building instructions)









### 3.3.3 Count lines

Make a program that counts black lines in a 50cm white desk as the vehicle passes above them. At the end show how many black lines have been counted in the brick's screen.



Execute the following code (the programs run parallel - at the same time-). Test different width black lines and distances between them.

```

when program starts
  set movement motors to A+B
  set movement speed to 20 %
  move straight (0) for 50 cm
  write count

when program starts
  set count to 0
  forever
    E wait until color is black
    change count by 1
    E wait until color is white
  
```

The vehicle starts to move straight for 50 cm and then shows the number of black lines that counted on brick's screen.

Set the counter to zero

A black line is counted when the robot changes its position from a black area to a white area.

## 3.3.4 Follow line (P Controller)

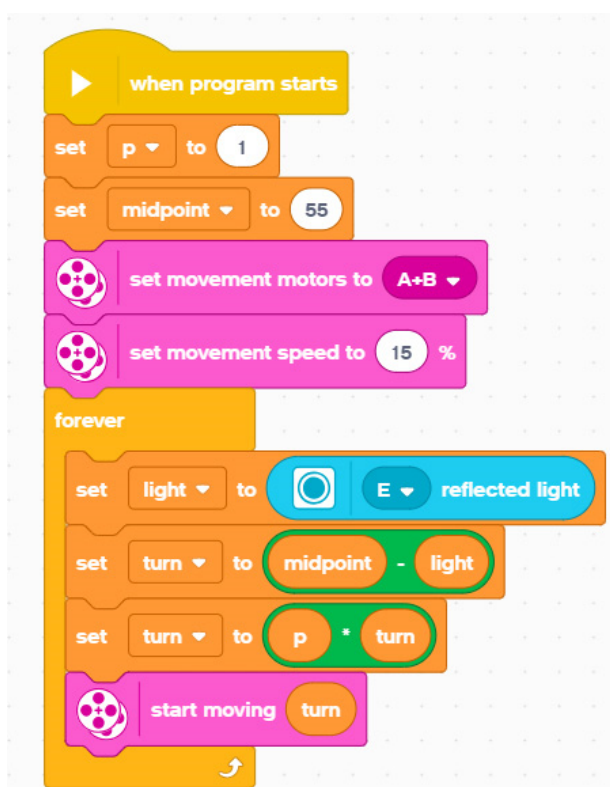
Make a program that helps the robotic vehicle to follow the edge of a black line for 20seconds.



Execute the following code (the programs run parallel - at the same time-).

**Some theory:** When the sensor is above the black area the value (BV) is small (almost zero) and when it is above the white area the value (WV) is very high (almost 100). The ideal follow line is for the sensor to be always in the middle  $((BV+WV)/2 = 50)$  - this is the midpoint-. When the sensor is above white area reads values above 50 and the vehicle must fix that error by turning it to the black area and vice versa. The turn value usually is the same as the error value but we can adjust it using a parameter called P (Proportion) that multiplies itself with the error value.

**Note:** Find your midpoint and test different movement speed and P values for a better and faster follow line.



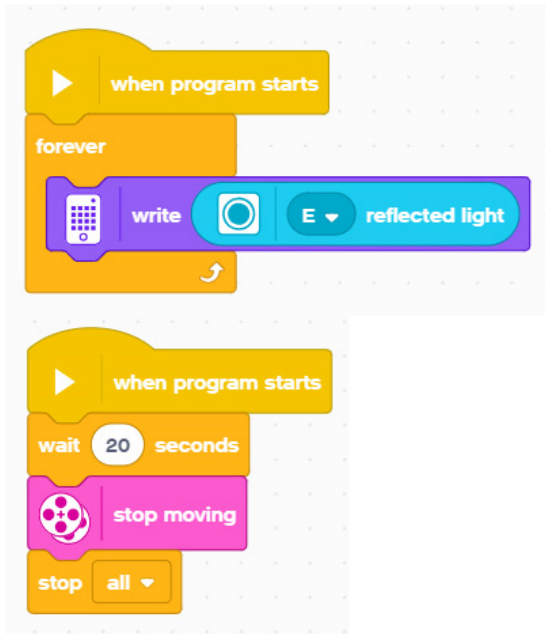
P value is set to 1

Midpoint value is set to 55

Light value is set to reflection value

The turn value is set to  $P * (\text{midpoint} - \text{light})$

The vehicle change it course by turn degrees



The reflection value appears on the brick' screen constantly.

The follow line program runs for 20 seconds

### 3.4 Gyro sensor



**Built-in  
6-axis gyro**

#### 3.4.1 Code blocks for gyro sensor

On the Sensors tab the gyro commands are shown below.



-*Command*. The block stops the execution of the program and continues when the value of the sensor is shaken, tapped or freefall.

-*Condition*. If the sensor's orientation is up it returns True.

-*Value*. Returns the value of the orientation (up, down, left side, right side, front or back).

-*Command*. Sets the yaw angle to zero.

-*Value*. Returns the value of the pitch, roll or yaw angle.

#### 3.4.2 Go straight ... no matter what!

Make a program that helps the robotic vehicle go straight even if someone tries to change it course. Display the yaw angle of the vehicle on brick's screen. The vehicle must stop when finds an obstacle in a small distance.

Execute the following code (the programs run parallel - at the same time-).

**Note:** Adjust the distance sensor add-on to the vehicle



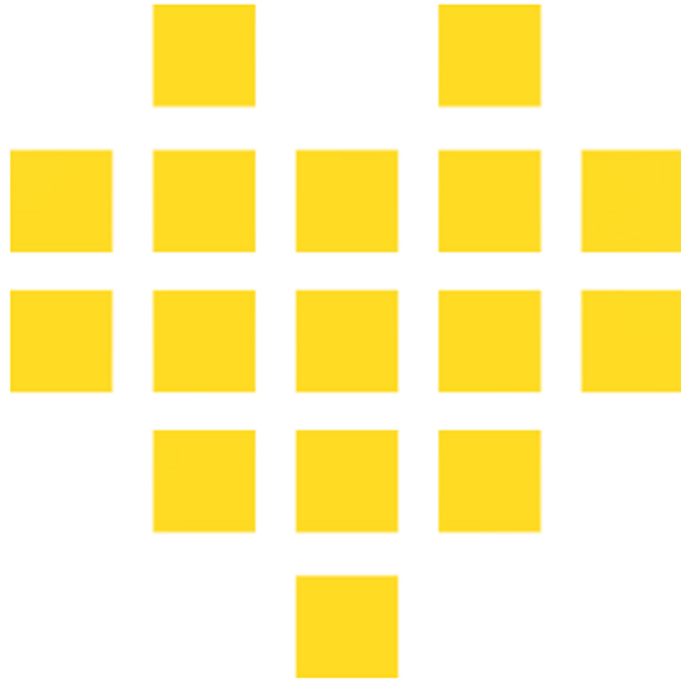
Set the yaw angle to zero.

When someone changes the course of the vehicle then it must turn to opposite direction (and that's why the minus is necessary).

The yaw angle value appears on the brick' screen constantly.

The program stops when an obstacle appears.





**ISBN: 978-618-00-1730-4**